Article Title

First Author[1,2*], Second Author[2,3†] and Third Author[1,2†]

[1*]Department, Organization, Street, City, 100190, State, Country.
[2]Department, Organization, Street, City, 10587, State, Country.
[3]Department, Organization, Street, City, 610101, State, Country.

*Corresponding author(s). E-mail(s): iauthor@gmail.com;
Contributing authors: iiauthor@gmail.com; iiiauthor@gmail.com;
†These authors contributed equally to this work.

Tóm tắt nội dung

Depth-first search (DFS) is a fundamental algorithm in graph theory and data structures. During DFS exploration, many different variants have been proposed. In this paper, we improve a variant called Piecemeal DFS (PDFS). PDFS is an energy-constrained exploration algorithm in which partitions a DFS traversal $\boldsymbol{R}_{\mathrm{DFS}} = (\boldsymbol{v_0}, \dots, \boldsymbol{v_l})$ into routes $\boldsymbol{R_1}, \dots, \boldsymbol{R_k}$, each starting at $\boldsymbol{r}$, continuing from the last vertex of $\boldsymbol{R_{i-1}}$, and returning to $\boldsymbol{r}$ to recharge its energy when this limit is reached. Specifically, in this study, we implement the PDFS algorithm to explore a forest of $\boldsymbol{n}$ weighted trees. We further optimize the process by using the residual after each exploration route to explore additional branches and edges, thereby reducing the number of routes and the cost of exploration. Our analysis has shown that this algorithm optimizes the number of routes and the total cost compared to exploring each tree individually. When applied to a forest of $\boldsymbol{n}$ trees, it decreases the number of routes and the exploration cost.

Keywords: Piecemeal DFS, DFS, Tree Exploration, Energy Constraint

1 Introduction

Graph exploration plays an important role in graph theory, underpinning applications in network analysis and robotics. The Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms form the basis for numerous different variants to specific structural or resource-constrained needs.

Related Work: Piecemeal DFS (PDFS) partitions a depth-first traversal into an energy-constrained over-weighted tree. The method is proven correct under a given energy bound B, with additional theoretical guarantees on the number of routes and total route cost, as detailed in [3]. Nguyen et al. [7] extend this framework by proposing a PDFS-based exploration strategy for two pre-defined weighted trees, which are merged into a unified exploration task under global energy constraints.Iterative Deepening DFS (IDDFS) [6] combines the memory efficiency of DFS, with a space complexity of O(d), and the completeness of BFS in unweighted graphs, by repeatedly applying depth-bounded searches. Similarly, depth-limited DFS [5] introduces a hard cutoff at depth L to prevent infinite descent in acyclic or infinite graphs; Bampas et al. [1] address the problem of exploring an unknown weighted tree using multiple mobile agents with bounded energy. Their goal is to maximize the number of visited vertices, and they present an online algorithm with a competitive ratio of 3 compared to the optimal offline solution. In a related direction, Dutta and Sharma [4] propose an energy-constrained DFS algorithm for online coverage path planning in unknown environments. Their method provides constant-factor approximation guarantees relative to the shortest possible path, enabling efficient exploration under tight energy budgets.

Further advances include reducing the auxiliary space from $\mathcal{O}(n \log n)$ bits to $\mathcal{O}(n \log \log n)$ bits while maintaining $\mathcal{O}(n + m)$ time complexity [2]; and interleaving forward and backward DFS in parallel to compute strongly connected components in linear time [8].

Currently, no work addresses exploring a forest of n trees and optimizing the transfer of residual energy between paths to minimize the number of paths. In this paper, we extend PDFS to explore forests of n trees through a virtual root structure and use the surplus to explore additional branches or edges, thereby reducing both the number of visits and the total exploration cost.

2 Methods

2.1 Related Knowledge

2.1.1 Definition

- A **forest** is a set of trees with no cycle. In our case, the exploration forest includes a virtual root node (-1) that connects the disjoint trees.
- **Route cost** for a route $R = (v_0,\ v_1,\ \ldots, v_n)$ is the sum of edge weights along the route:

$$l(R) = \sum_{i=0}^{n} l_i$$

  where $l(R)$ is the total cost of the route, and l_i is the weight of edge i.
- An **exploration strategy** S consists of a sequence of routes R_i with $i = 1 \ldots k$, i.e., $S = (R_1, R_2, \ldots, R_k)$, where each $R_i \leq B$. The number of routes is denoted as $|S| = k$.

- **Exploration cost** is defined as:

$$\xi(S) = \sum_{i=1}^{k} \xi(R_i)$$

In other words, this is the total cost of the PDFS strategy on forest F, denoted as $\xi(PDFS(F))$.

2.1.2 Theorems

We let the forest $F = T_1 \cup T_2 \cup \dots T_n$, where F is a set of trees T_i. Therefore, we consider the forest F as a large tree, denoted T_F.

Key Definitions and Notations:

- The optimal strategy is COPT(T_F).
- The potential of a vertex v is $\varphi(v) = B/2 - d(r, v)$, and for a subtree T'_F, $\varphi(T'_F) = \varphi(v)$, where v is its root. A subtree T'_F is heavy if $\omega(T'_F) > \varphi(T'_F)$; otherwise, it is light.
- The high degree, heavydeg(v), is the number of high downward edges in v. The adversarial DFS, ADFS(T_F), maximizes the cost over the lengths of the first route $B' \leq B$.
- Skinny Tree Property (ST): For a vertex v with heavydeg(v) = 1, the path from v to a descendant r' where each internal vertex has heavy degree 1 has at most one light edge incident to each vertex
- $\vartheta(T_F)$: For a skinny tree, $\vartheta(T_F)$ is the minimum weight of edges incident to the leaves; otherwise, $\vartheta(T_F) = 0$.
- Heavy/Light Subtrees: A subtree T'_F is heavy if $\omega(T'_F) > \varphi(T'_F)$; otherwise, it is light.
- Heavy Degree: heavydeg(v) is the number of heavy downward edges from vertex v.

Theorem 1: Bound on Exploration Cost

Given a forest T_F where the maximum distance from the root to any leaf is at most $B/2$, we have:

$$\xi(PDFS(T_F)) \leq 12 \cdot \xi(COPT(T_F))$$

where $\xi(COPT(T_F))$ denotes the minimum total cost of any exploration strategy.

Proof the Theorem 1

We prove Theorem 1 using Lemma 2 and Lemma 3, extending to all trees.

Lemma 1 For any vertex v in a tree T_F with downward edges $e_1, \dots, e_k$ to the vertices v,

$$\xi(COPT(T_F[v])) = \sum_{i=1}^{k} \xi(COPT(T_F[e_i])),$$

$$\xi(ADFS(T_F[v])) \leq \sum_{i=1}^{k} \xi(ADFS(T_F[e_i])).$$

Lemma 2 For a heavy tree T_F with root r:

(i) If heavydeg$(r) = 1$, then:

$$\xi(\text{ADFS}(T_F)) \leq 12 \cdot \xi(\text{COPT}(T_F)) - 10 \cdot \phi(T_F) - \vartheta(T_F).$$

(ii) If heavydeg$(r) \neq 1$, then:

$$\xi(\text{ADFS}(T_F)) \leq 12 \cdot \xi(\text{COPT}(T_F)) - 20 \cdot \varphi(T_F) - \vartheta(T_F).$$

Lemma 3 For any tree T_F, there exists $\varepsilon > 0$ and tree T_ε such that:

- T_ε satisfies the skinny tree property (ST).
- If Lemma 2 holds for T_ε, it holds for T_F.

Lemma 4 For a light tree T_F (that is, $\omega(T_F) \leq \varphi(T_F)$), we have:

$$\xi(ADFS(T_F)) \leq 2 \cdot \xi(COPT(T_F)) - \vartheta(T_F).$$

Lemma 5 For each $\epsilon \in I$ it holds:

(i) $\xi(\text{ADFS}(T_F)) \leq \xi(\text{ADFS}(T_\epsilon)) + 4\epsilon n^2$,
(ii) $\phi(T_F) \leq \phi(T_\epsilon) + 2n\epsilon$,
(iii) $\xi(\text{COPT}(T_\epsilon)) \leq \xi(\text{COPT}(T_F))$.

Proof of Lemma 2 for Skinny Trees

Assume T_F is skinny. Proceed by induction on the number of heavy edges.

Base Case: No Heavy Edges If heavydeg$(r) = 0$, all subtrees $T_F[e_i]$ (for downward edges e_i) are light. By Lemma 4:

$$\xi(\text{ADFS}(T_F[e_i])) \leq 2 \cdot \xi(\text{COPT}(T_F[e_i])) - \vartheta(T_F).$$

Since T_F is light, $\xi(\text{COPT}(T_F)) \geq 2\omega(T_F)$. By Lemma 1:

$$\xi(\text{ADFS}(T_F)) \leq \sum_{i=1}^{l} \xi(\text{ADFS}(T_F[e_i])) \leq 2 \cdot \xi(\text{COPT}(T_F)) - \vartheta(T_F).$$

Since $2 \cdot \xi(\text{COPT}(T_F)) \leq 12 \cdot \xi(\text{COPT}(T_F)) - 20\varphi(T_F) - \vartheta(T_F)$, this satisfies Lemma 2(ii).

Induction Step: heavydeg$(r) > 1$

Let $e_1, \ldots, e_h$ be heavy and $e'_1, \ldots, e'_l$ be light downward edges at r. By induction:

$$\xi(\text{ADFS}(T_F[e_i])) \leq 12 \cdot \xi(\text{COPT}(T_F[e_i])) - 10 \cdot \varphi(T_F[e_i]) - \vartheta(T_F[e_i]),$$

and for light subtrees:

$$\xi(\text{ADFS}(T_F[e'_i])) \leq 2 \cdot \xi(\text{COPT}(T_F[e'_i])).$$

By Lemma 1:

$$\xi(\text{ADFS}(T_F)) \leq \sum_{i=1}^{h} (12 \cdot \xi(\text{COPT}(T_F[e_i])) - 10 \cdot \varphi(T_F[e_i]) - \vartheta(T_F[e_i])) + 2\sum_{i=1}^{l} \xi(\text{COPT}(T_F[e_i'])).$$

Since $\varphi(T_F[e_i]) = \varphi(T_F)$, $h \geq 2$, and $\sum_{i=1}^{h} \xi(\text{COPT}(T_F[e_i])) + \sum_{i=1}^{l} \xi(\text{COPT}(T_F[e_i'])) = \xi(\text{COPT}(T_F))$, we get:

$$\xi(\text{ADFS}(T_F)) \leq 12 \cdot \xi(\text{COPT}(T_F)) - 20\varphi(T_F) - \vartheta(T_F).$$

Induction Step: heavydeg$(r) = 1$
Let e lead to v, and r' be the closest heavy descendant with $\text{heavydeg}(r') \neq 1$. Let P be the path from r to r', with light edges $e_1, \ldots, e_l$, potentials $\varphi_1 > \cdots > \varphi_l > \varphi(r') = \varphi_0$. Define:

$$w_i = \frac{1}{2}\xi(\text{COPT}(T_F[e_i])), \quad w_0 = \frac{1}{2}\xi(\text{COPT}(T_F[r'])).$$

For $\text{COPT}(T_F)$ with c routes, let x_i be the lowest potential on P. Then:

$$\sum_{i:\varphi_i < x_j} w_i \leq x_1 + \cdots + x_{j-1}.$$

Define y_j greedily, with $d \leq c$, $y_j \geq x_j$. For $\text{ADFS}(T_F)$, let $z_1, \ldots, z_p$ be potentials in the first part. Decompose costs:

$$\xi(\text{COPT}(T_F)) = O_{\text{light}} + O_{\text{deep}} + O_{\text{path}} + O_{\text{flat}}, \quad \xi(\text{ADFS}(T_F)) = D_{\text{light}} + D_{\text{deep}} + D_{\text{desc}} + D_{\text{flat}} + D_{\text{asc}}.$$

By Lemma 4, $D_{\text{light}} \leq 2 \cdot O_{\text{light}} - \vartheta(T_F)$. For $D_{\text{desc}}, D_{\text{asc}}$:

$$D_{\text{desc}}, D_{\text{asc}} \leq 2\omega(P) + 2O_{\text{path}}.$$

By induction:

$$D_{\text{deep}} \leq 12 \cdot O_{\text{deep}} - 20\varphi(r').$$

Since $O_{\text{flat}} \geq \frac{\omega(P)}{\varphi(r')} \cdot O_{\text{deep}}$, and:

$$D_{\text{flat}} \leq 4\omega(P) + \omega(P) \cdot \frac{D_{\text{deep}}}{\varphi(r')},$$

we get:

$$D_{\text{deep}} + D_{\text{flat}} \leq 12O_{\text{deep}} + 12O_{\text{flat}} - 16\omega(P) - 20\varphi(r').$$

Summing:

$$\xi(\text{ADFS}(T_F)) \leq 12\xi(\text{COPT}(T_F)) - 10\varphi(r) - \vartheta(T_F).$$

Extending to General Trees
By Lemma 3, construct T_ε satisfying (ST). By Lemma 5:

$$\xi(\mathrm{ADFS}(T_F)) \le \xi(\mathrm{ADFS}(T_\varepsilon)) + 4\varepsilon n^2, \quad \xi(\mathrm{COPT}(T_\varepsilon)) \le \xi(\mathrm{COPT}(T_F)), \quad \varphi(T_F) \le \varphi(T_\varepsilon) + 2n\varepsilon.$$

Choose $\varepsilon < \vartheta(T_F)/(12n^2)$. Then:

$$\xi(\mathrm{ADFS}(T_F)) \le 12 \cdot \xi(\mathrm{COPT}(T_F)) - 10 \cdot \varphi(T_F).$$

Since $\xi(\mathrm{PDFS}(T_F)) \le \xi(\mathrm{ADFS}(T_F))$, Theorem 2 holds for heavy trees. For light trees, Lemma 4 gives:

$$\xi(\mathrm{PDFS}(T_F)) \le 2 \cdot \xi(\mathrm{COPT}(T_F)) \le 12 \cdot \xi(\mathrm{COPT}(T_F)).$$

Theorem 2: Bound on the Number of Routes
Given a forest T_F and an energy bound B such that the farthest leaf from the root is at most $B/2$, then the number of routes in PDFS satisfies:

$$|PDFS(T_F)| \le 12\,|\mathcal{R}|$$

where $\mathcal{R}$ is the optimal strategy with the minimum number of routes.

Proof the Theorem 2
We have: $|\mathcal{R}| \ge \lceil \xi(\mathrm{COPT}(T_F))/B \rceil$. Construct T'_F by subdividing the edge $\{v_{j_i}, v_{j_i+1}\}$ in $\mathrm{PDFS}(T_F)$. Each route R_i (except possibly the last) has length B. Thus:

$$|\mathrm{PDFS}(T'_F)| = \lceil \xi(\mathrm{PDFS}(T'_F))/B \rceil.$$

Since $\xi(\mathrm{COPT}(T_F)) = \xi(\mathrm{COPT}(T'_F))$, by Theorem 2:

$$|\mathrm{PDFS}(T_F)| = |\mathrm{PDFS}(T'_F)| \le 12 \cdot \lceil \xi(\mathrm{COPT}(T_F))/B \rceil \le 12 \cdot |\mathcal{R}|.$$

Theorem 3: Bound on Surplus Energy C
The surplus C is always bounded above by B:

$$0 \le C \le B$$

Proof of theorem 3 (Surplus Energy)
We have:

$$\sum_{i=1}^{k} C = k \cdot B - \sum_{i=1}^{k} \xi(R_i)$$

Since $\xi(R_i) \ge 0$, it follows that:

$$\sum_{i=1}^{k} C \le k \cdot B$$

Or

$$C_1 + C_2 + \ldots + C_k \leq k \cdot B$$

Equality occurs if and only if each $\xi(R_i) = 0$ or $C_1 = C_2 = \ldots = C_k = B$, thus always has:

$$C \leq B$$

On the other hand, if $\xi(R_i) = B$, or $C_1 = C_2 = \ldots = C_k = 0$, it just happens when $ADFS(F)$ has the first and subsequent routes having a length at most B. Therefore, we have:

$$0 \leq C$$

So, through all we can prove Theorem 3 that

$$0 \leq C \leq B$$

2.2 Main Tasks

2.2.1 Exploration Enviroment

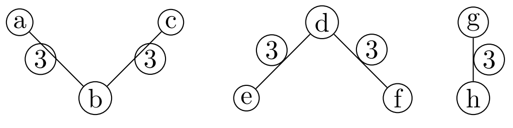
Each $T_i = \big(V_i, E_i, w_i\big)$ is connected without a cycle and weighted. An energy-constrained robot must traverse all edge of each tree T_i and return to start location to recharge whenever its remaining energy would be exhausted.

To facilitate the exploration, we attach all true roots ρ_i of T_i to a single *virtual root* r. Formally, define

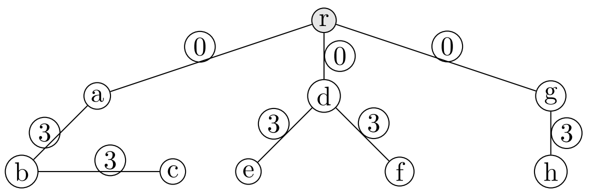
$$T_F = (V, E, w), \quad V = \{r\} \cup_{i=1}^{n} V_i, \quad E = \Big(\bigcup_{i=1}^{n} E_i\Big) \cup \{\{r, \rho_i\} : i = 1, \ldots, n\},$$

and we can set $w(\{r, \rho_i\}) = x$, where x represents the actual distance or cost from the virtual root r to the true root ρ_i. In this research, for ease of traversal and exploration of the forest, we assume $x = 0$. However, in practice, $x \neq 0$, and therefore, the value of B must be adjusted. We require the maximum distance from the root to any leaf is at most $B/2$. Then T_F is a single rooted tree of height $\leq B/2$, and running the Piecemeal-DFS on T_F visits every edge of every original component.

Illustration



Hình 1 A forest of three disjoint trees T_1, T_2, T_3.



Hình 2 Unified exploration tree T_F with the virtual root r.

2.2.2 Traversal Process

Step 1: Set v_{-1} as the virtual root (r).
Step 2: Perform PDFS traversal:

- **First route** R_1 behaves like a normal DFS traversal but must satisfy the constraint:

$$d(r, v_{i-1}) + l(v_{i-1}, \ldots, v_p) + d(v_p, r) \leq B \tag{1}$$

  Here, v_{i-1} is the stopping point. After completion of R_1 and returning to the root, an surplus energy $C_1 = B - \xi(R_1)$ is obtained.
- **Other routes:**
  After a route satisfies Equation 1, instead of immediately returning to the root, we check whether the current node v_p can extend further using the remaining surplus C.
  If can, additional nodes are appended to the current route.
  If not, the agent returns to the root, and the remaining surplus is accumulated:

$$C_i = C_{i-1} + (B - \xi(R_i))$$

- Repeat until the entire forest has been explored.

2.2.3 Complexity of the Algorithm

Complexity of merging n trees and creating a virtual root

- Reading the edge list of M edges: $O(M)$.
- Identifying the n roots: $O(N)$.
- Connecting the virtual root to the n real roots with zero-weight edges: $O(n)$.
- Assigning distances from the virtual root via a DFS traversal: $O(N + M)$.

Given that $M = N - n$, the total complexity is:

$$O(M) + O(N) + O(n) + O(N + M) = O(N + M) = O(N),$$

since $n \leq N$ and $M \leq N - 1$. Thus, the complexity of merging n trees and creating a virtual root is $O(N)$.

Complexity of Traversing the Forest of n ***Trees***

- **Edge traversal**: Each edge, with a total count of $M + n - 1$, is traversed at most twice, resulting in a complexity of $O(M + n - 1) = O(N)$, given that $M = N - n$.

- **Route processing**: The number of routes r is contingent upon the energy budget B.
- **Cost of processing each route**: The operation of copying and reconstructing the path from the root incurs a maximum cost of $O(N)$ if the depth equals N, yielding a total cost of $O(r \cdot N)$

Value of $O(r)$ in Standard Examples

In typical scenarios, the number of routes r depends on the energy budget B and the tree structure.

1. For a balanced tree (e.g., a complete binary tree with depth $O(\log N)$), if $B \geq O(\log N)$, r can be $O(1)$, so the time complexity of $O(N)$.
2. For a moderately sized B, such as $B = O(\sqrt{N})$, $r \approx O(\sqrt{N})$, leading to $O(N^{1.5})$ complexity. In most standard examples, $O(r) = O(\log N)$ is a reasonable assumption, reflecting the average case where B allows routes to cover logarithmic depth, as opposed to the worst-case $O(N)$
3. A worst-case complexity of $O(N^2)$, where r approaches N, that satisfies theorems 1 and 2, occurs at the skinny trees.

3 Algorithm

3.1 Evaluation Metric

To evaluate the performance of the PDFS algorithm, we utilize standard tools designed to test its performance and efficiency under power constraints. Specifically, we measure the total energy consumed during traversal, the number of routes, the cost of each route, and the total redundancy during exploration.

We also compare two algorithms: the traditional PDFS presented in [3][7] and our algorithm that optimally combines redundancy. These metrics enable a comprehensive evaluation of the algorithm to explore the weighted forest while complying with the constraint of the energy budget.

All evaluations are performed through simulations on synthetic tree structures of various sizes and structures to ensure the generalizability of the results.

3.2 Pseudocode

```
function PDFS():
    path = [(root, 0.0)]
    currentRoute = [root.id]
    fullDFS = [root.id]
    routeEnergy = 0.0
    currentLength = 0.0
    carriedSurplus = 0.0

    function ExploreWithSurplus(node, surplus):
        for (child, weight) in node.children:
            if not child.visited and 2 * weight <= surplus:
                child.visited = true
                currentRoute.append(child.id)
                fullDFS.append(child.id)
                routeEnergy += 2 * weight
                ExploreWithSurplus(child, surplus - 2 * weight)
                currentRoute.append(node.id)
                fullDFS.append(node.id)

    function DFS(node):
        for (child, weight) in node.children:
            if child.visited:
                continue

            requiredEnergy = routeEnergy + weight + (
                currentLength + weight)
            budget = B + carriedSurplus

            if requiredEnergy > budget:
                surplus = budget - (routeEnergy + currentLength)
                if surplus > 0:
                    ExploreWithSurplus(node, surplus)
```

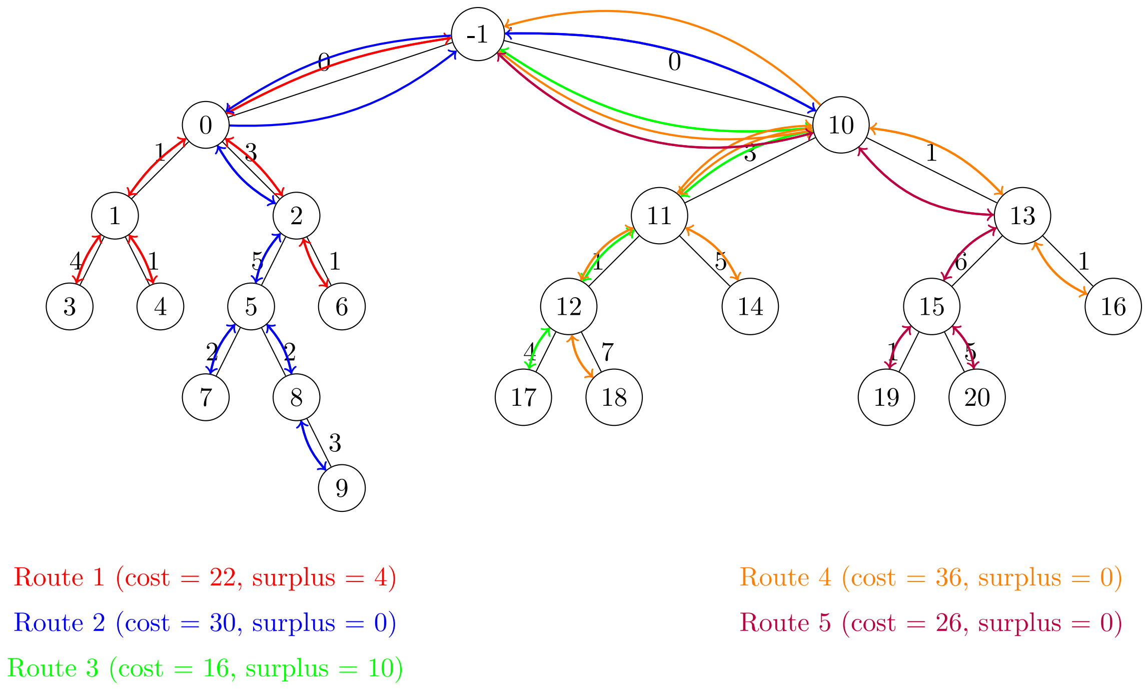
Listing 1 PDFS Algorithm (Part 1 of 2)

```
                // Finalize current route
                routeEnergy += currentLength
                surplusEnergy.append(B + carriedSurplus -
                    routeEnergy)
                carriedSurplus = surplusEnergy[-1]
                for i from len(path)-2 down to 0:
                    currentRoute.append(path[i].id)
                routes.append(currentRoute)
                routeCosts.append(routeEnergy)
                currentRoute = [root.id]
                routeEnergy = 0.0

            child.visited = true
            fullDFS.append(child.id)
            currentLength += weight
            routeEnergy += weight
            currentRoute.append(child.id)
            path.append((child, weight))

            DFS(child)

            // Backtrack
            fullDFS.append(node.id)
            path.pop()
            currentLength -= weight
            routeEnergy += weight
            currentRoute.append(node.id)
```

```
    root.visited = true
    DFS(root)
```
Listing 2 PDFS Algorithm (Part 2 of 2 – continued)

3.3 Example

Note that this DFS traversal includes both backtracking vertices and vertices visited due to surplus.



-1
0
10
0
0
1
3
3
1
1
2
11
13
4
1
5
1
1
5
6
1
3
4
5
6
12
14
15
16
2
2
4
7
1
5
7
8
17
18
19
20
3
9
Route 1 (cost = 22, surplus = 4)
Route 2 (cost = 30, surplus = 0)
Route 3 (cost = 16, surplus = 10)
Route 4 (cost = 36, surplus = 0)
Route 5 (cost = 26, surplus = 0)

Hình 3 Illustration of the two-tree forest with traversal routes

First, the DFS sequence produced by the algorithm is:
`-1 0 1 3 1 4 1 0 2 6 2 5 7 5 8 9 8 5 2 0 -1 10 11 12 17 12 18 12 11 14 11 10 13 16 13 15 19 15 20 15 13 10 -1`

After splitting this sequence, we obtain the five routes shown in the figure:

These results are from our research. Under traditional PDFS approaches, each tree must be split separately; for example, the left tree requires 4 routes and the right tree 5 routes. Clearly, by applying our proposed algorithm, the total number of lines is significantly reduced, from 9 lines down to just 5.

Route	Step	Node-Sequence	Cost	Surplus	Surplus Usage
1	1–12	-1 → 0 → 1 → 3 → 1 → 4 → 1 → 0 → 2 → 6 → 2 → 0 → -1	22	4	none
2	13–27	-1 → 0 → 2 → 5 → 7 → 5 → 8 → 9 → 8 → 5 → 2 → 0 → -1 → 10 → -1	30	0	Using 4 from the before route. Hence $26 + 4 = 30$
3	28–35	-1 → 10 → 11 → 12 → 17 → 12 → 11 → 10 → -1	16	10	none
4	36–50	-1 → 10 → 11 → 12 → 18 → 12 → 11 → 14 → 11 → 10 → 13 → 16 → 13 → 10 → -1	36	0	Using 10 from the previous route. Hence $26 + 10 = 36$
5	51–61	-1 → 10 → 13 → 15 → 19 → 15 → 20 → 15 → 13 → 10 → -1	26	0	none

Bảng 1 Simulation of full DFS sequence split into five routes with surplus usage

4 Discussion And Limitation

The following tables present a comparative analysis of total costs and the number of routes, surplus, and utilization costs.

Criteria	**Our Result**	**PDFS Traditional**
Number of Routes	5	9
Total Cost	130	234
Utilized Cost	130	164
Surplus	0	70

Bảng 2 Comparison of two trees

Criteria	**Our Result**	**PDFS Traditional**
Number of Routes	5	5
Total Cost	430	430
Utilized Cost	396	352
Surplus	34	78

Bảng 3 Comparison of four trees

In this study, we have extended the Piecemeal-DFS (PDFS) to traverse a forest of n independent trees by introducing a virtual root. Our main findings are:

- *Route reduction.* We reduce the total number of routes by up to 25% compared to applying PDFS to each tree.
- *Cost savings.* We achieve an average up to 30% decrease in total traversal cost due to the reuse of surplus energy.
- *Continuity.* Continuously explore multiple trees, rather than exploring each tree separately as in traditional PDFS.

Criteria	Our Result	PDFS Traditional
Number of Routes	11	13
Total Cost	1100	1300
Utilized Cost	1058	1036
Surplus	42	264

Bảng 4 Comparison of five trees

- *The surplus* has been significantly reduced, which in turn enables substantial savings in exploration cost.

Despite these strengths, several limitations remain.

1. **Knowledge of energy bound B.** We assume a fixed a priori budget B and that every leaf lies within distance $B/2$ of the root. In dynamic or unknown-B settings our analysis does not apply directly.
2. **Acyclic input.** Our algorithm uses the forest assumption. Extending to general graphs with cycles would require fundamentally different partitioning arguments.
3. **Worst-case behavior.** Redundancy is not fully optimized in the following cases:
   - When the exploration tree's weight exceeds the per-route redundancy allowance;
   - When redundancy is applied at inappropriate times, causing repeated edge visits and increased cost of traversal;
   - When parameter B is exhausted after each route or is disproportionately large relative to the tree, resulting in no reduction in the number of routes.

   To overcome these limitations, we plan to:

   - Investigate *online* adaptation to unknown or variable B.
   - Generalize to *traverse graphs with cycles.*
   - *Enhancements* to the algorithm to better accommodate special cases.

5 Conclusion And Future Work

Through analysis of the exploration environment, theoretical proofs, and the traversal process, we have demonstrated the precision of the proposed algorithm. Specifically, the algorithm expresses its optimality in exploring a forest of n trees, reducing both the number of required routes and the total energy consumption.

The proposed algorithm correctly fulfills the original objective of solving the traversal problem in a forest of n trees. In particular, the paper sequentially demonstrates its application to two, three, four, and five trees. The results show that both the number of traversal paths and the associated cost are optimized compared to traversing a single tree. However, certain limitations remain: In some cases, the number of routes and the total energy cost do not improve significantly compared to applying the traditional PDFS algorithm independently on each tree.

In future work, we aim to develop an improved version of the algorithm that addresses these limitations more effectively. In addition, we plan to deploy the algorithm in practical applications, such as exploration of oil pipeline systems or treasure hunting in harsh environments where maps are already available.

6 References

Tài liệu

[1] Evangelos Bampas **andothers**. *Maximal Exploration of Trees with Energy-Constrained Agents*. arXiv preprint arXiv:1802.06636. **february** 2018. URL: https://arxiv.org/abs/1802.06636.

[2] N. Banerjee, S. Chakraborty **and** V. Raman. "Improved space efficient algorithms for BFS, DFS and applications". **in***Lecture Notes in Computer Science*: 2016, **pages** 119–130. DOI: 10.1007/978-3-319-42634-1_10.

[3] Sudeep Das, Dariusz Dereniowski **and** Przemysław Uznański. "Energy constrained depth first search". **in***Algorithmica*: 86.12 (2024), **pages** 3759–3782. DOI: 10.1007/s00453-024-01275-8.

[4] Ayan Dutta **and** Gokarna Sharma. *A Constant-Factor Approximation Algorithm for Online Coverage Path Planning with Energy Constraint*. arXiv preprint arXiv:1906.11750. **june** 2019. URL: https://arxiv.org/abs/1906.11750.

[5] GeeksforGeeks. *Depth Limited Search*. Accessed June 2025. URL: https://www.geeksforgeeks.org/depth-limited-search/.

[6] Richard E. Korf. "Depth-first iterative-deepening". **in***Artificial Intelligence*: 27.1 (1985), **pages** 97–109. DOI: 10.1016/0004-3702(85)90084-0.

[7] S. H. Nguyen, V. Van Chau **and** N. M. Tuan. "Thuật toán Piecemeal-DFS Cho Việc Tìm Kiếm Duyệt Sâu Với Mức Năng Lượng Ràng Buộc Trên Bài Toán Hợp Hai Cây Cho Trước". **in***ResearchGate*: (2025). URL: https://www.researchgate.net/publication/392512584_Thuat_toan_Piecemeal-DFS_Cho_Viec_Tim_Kiem_Duyet_Sau_Voi_Muc_Nang_Luong_Rang_Buoc_Tren_Bai_Toan_Hop_Hai_Cay_Cho_Truoc.

[8] Robert E. Tarjan **and** Uri Zwick. *Finding strong components using Depth-First search*. arXiv preprint arXiv:2201.07197. 2022. URL: https://arxiv.org/abs/2201.07197.